For-Loops

Introduction to Computer Programming

Dr. Paul Vrbik

September 28, 2018

String Methods

Dozens of built-in (i.e. usable on Assignment 1) string methods can be found here

`docs.python.org/3/library/stdtypes.html#string-methods`

```
>>> "steven universe".capitalize()
'Steven universe'
>>> "steven universe".count("e")
4
>>> "steven universe".find("e")
2
```

String Methods

```
>>> "123".isdigit()
True
>>> "123.456".isdigit()
False
>>> "123.456".find(".")
3
>>> "123.456E7".find("E")
7
```

String Methods

```
>>> ord('a')
97
```

Because `a` *is Unicode character 97.*

```
>>> chr(97)
'a'
```

```
>>> chr(128512)
:)
```

String Methods

```
>>> chr( ord('a')+2 )
'c'


>>> 'a' > 'A'
True


>>> ord('a') > ord('A')
True


>>> ord('a') - ord('A')
32
```

Remember this number.

Question

Write a function that converts a lower case character to its capital.

Answer

```
def to_capital(c:str) -> str:
    """Assumes input is a lower case character and
    converts it to its capital.
    >> to_capital('k')
    'K'
    """
    return chr( ord(c) - 32 )
```

Question

Write a function that converts all the lower case characters in a string to its capital.

Answer

```
def to_capitals(cs:str) -> str:
    """Converts each character of a string to its capital.
    >>> to_capitals("kAngaRoO")
    'KANGAROO'
    >>> to_capitals("DROP bear")
    'DROP BEAR'
    >>> to_capitals("123go")
    '123GO'
    """
```

Definition (Loop)

A loop is a control structure that repeats code that belongs to it.

Definition (For-Loop)

A for-loop is a control structure that, given a group, repeats code for every member that group in order.

Definition (For-Loop)

```
for <name> in <iterator>:
    <code>
```

```
>>> for x in "abcd":
...     print(x)
a
b
c
d
>>> x
'd'
```

Notice the order.

Nesting For-Loops

```
>>> (digits, alphas) = ("012", "ab")
... for d in digits:
...         for a in alphas:
...             print(d + a)
0a
0b
1a
1b
2a
2b
```

Nesting For-Loops

```
>>> (digits, alphas) = ("012", "ab")
... for d in digits:
...         for a in alphas:
...             None        (Python disallows empty for-loops.)
...         print(d + a)
0b
1b
2b
```

Nesting For-Loops

```
>>> (digits, alphas) = ("012", "ab")
... for d in digits:
...         for a in alphas:
...             None
... print(d + a)
2b
```

Note (again) the names used to iterate through the iterator retain their value after the for-loop exists.

Definition (Accumulator)

An accumulator is a variable which a loop uses to 'accumulate' an aggregate value.

```
>>> acc = ""
>>> for x in "abcd":
...     acc = acc + x
...     print(acc)
a
ab
abc
abcd
```

Question

Write a function that reverses a string when given a string.

Answer

See `reverse_string.py` on course web-site.

Returning to. . .

Question

Write a function that converts a lower case character to its capital.

Answer

See `upper_case.py` on course web-site.

Definition (Caesar cipher)

A Caesar cipher is a type of encryption where each letter in the plaintext is 'shifted' a certain number of places down the alphabet to obtain the ciphertext.

For instance, with a shift of -2 we do

$$A \to Y$$

$$B \to Z$$

$$C \to A$$

$$\vdots$$

$$Z \to X$$

Question

Write code for encrypting and decrypting messages using the Caesar cipher.

In particular, write functions with the headers

```
def encrypt_caeser(plaintext:str, shift:int) -> str:
```

```
def decrypt_caeser(ciphertext:str, shift:int) -> str:
```

Answer

See `caesar.py` on course web-site.

Next Time

1. More accumulation.
2. Condition checking.